EVONET'03

EVOLVING A ROBOT NEUROCONTROLLER FOR MEMORIZATION TASKS

E. A. Buskum, S. Gervasi, R. Gross, B. De Vylder, S. Yildirim

Abstract:

In this study, a robotic neurocontroller architecture that handles delayed response tasks, which are characterized by a significant (and typically varying) delay between a stimulus and the corresponding appropriate response, is proposed. That is, the architecture aims at memorization and counting tasks. Current controller architectures use supervised (eg. gradient descent), reinforcement learning or evolutionary algorithms for this class of learning tasks. However, gradient descent methods have serious limitations such as being unable to handle memory very well, whereas supervised and reinforcement learning require a priori knowledge which might not always be available. Approaches for evolution of ANN as a controller includes evolution of initial weights, evolution of learning rules and evolving fixed weights. In this paper, initial and fixed weights are used for evolution of response and delay tasks.

Keywords

Robotic neurocontroller, evolution, delayed response tasks, stimulus, counting.

1. Introduction

In the past, experiments relating to memorization and counting tasks have been conducted in diverse research areas.

Reynikova & Ryabko (2000) has a study about how ants communicate some information to each other when they find new places of food in corridors of simple mazes. The ants used in their experiments are not allowed to follow each other to learn the place of food discovered by another ant. They were not able to detect pheromone tracks either.

Boden, Jacobsson & Ziemke (2000) has a study on predicting the elements of a binary sequence with evolved neural networks. This task required the counting of bits. The recurrent neural networks and inputs were updated synchronously.

The previous two studies and the ones in the following paragraphs, achieved the task of "counting to 2". These are the main motivations for our research on *evolving* a robot neurocontroller for tasks of counting to three or more.

In 1996, Ulbrich used two networks (a) The input state and (b) Elman (1990) to solve the T-maze problem. Figure 1. shows the two situations in a typical T-maze problem. The task of the robot is defined as taking a right or a left turn at the T-maze junction depending on the stimulus, a light, it encounters either on the left or the right of the corridor of a T-maze towards its junction. The robot has to 'remember' where it has seen the light, because of the delay between the stimulus and the appropriate response.

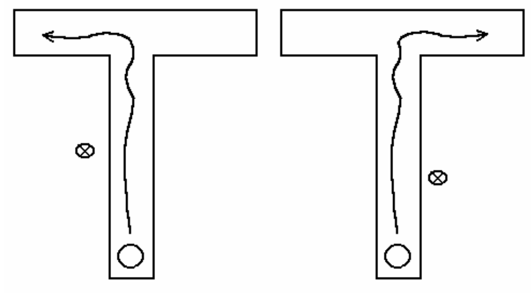

Figure 1. The two situations in the simple T-maze environment. Adapted from Ulbricht (1996).

In her study, both networks were trained using backpropagation and there was a constant delay between the stimulus and the junction. The input state network learned this task correctly and also displayed a capacity to generalize the delay to longer and shorter time lags. The network was eighty percent successful. On the other hand, the Elman network (Figure 2) failed to solve the task. Ulbricht (1996) points out that this may be due to the learning algorithm rather than an inherent weakness of the architecture (e.g. Ulbricht, 1996; Jakobi, 1997; Rylatt & Czarnecki, 2000; Linåker & Jacobsson, 2001).

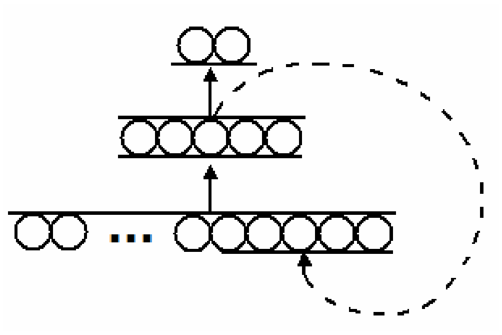

Figure 2. Elman network. Adapted from Ulbricht.

Later, in 2000, Rylatt and Czarnecki modified the Elman (1990) network with supervised learning in order to see if that could improve the poor results obtained by Ulbricht (1996) with respect to the T-maze environment. The simulated environment used in this experiment, illustrated in Figure 3, consists of a corridor with an alcove that can appear either in the right or in the left wall. The robot makes turns depending on where the alcove is. The modified Elman network (Figure 4) is shown to have some capacity to deal with this tasks contrary to Ulbricht's (1996). results.

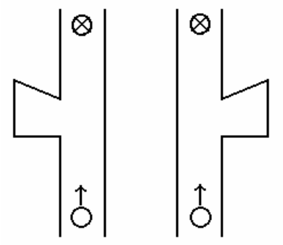

Figure 3. The simulated environment.

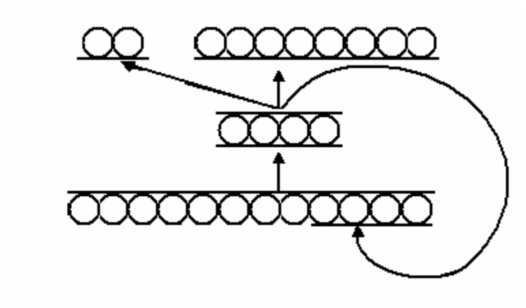

Figure 4. The modified Elman network. Adapted from Rylatt and Czarnecki (2000).

In the meantime, Jakobi (1997a, 1997b) used evolutionary algortihms in contrary to earlier work as a solution to the T-maze environment. He used self-organized recurrent network. The fitness was taken to be the total distance traveled in the two corridors plus a large score if the agent turned to the correct direction.

In addition to the T-maze problem explained above, Bergfeldt & Linåker (2002), worked on other tasks called "repeated T-maze" and "multiple T-maze" respectively. In repeated T-maze (Figure 5) the robot turns either right or left at each junction depending on where it has seen a stimulus (on the right or left) in the recent corridor.

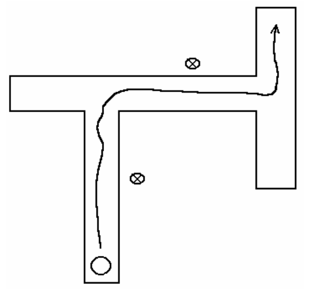

Figure 5. Repeated T-maze

Figure 6 shows a multiple T-maze where a robot meets all the stimuli in the first corridor and has to recall the corresponding stimulus at each junction.

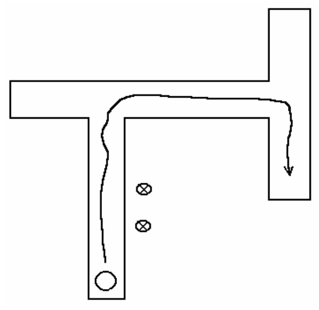

Figure 6. Multiple T-maze

Bergfeldt & Linåker (2002) tested each of Feedforward Network, Recurrent Network, Sequential Cascaded Network (SCN) and Extended Sequential Cascaded Network (ESCN) for each of repeated and multiple T-maze tasks. They achieved an average of %99.5 success with ESCN (Figure 7). That is, they were able to count up to *two*.

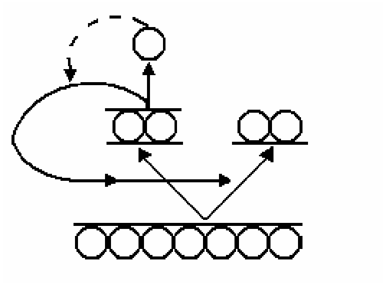

Figure 7. Extended Sequential Cascaded Network (ESCN).

The neural network arhictectures tested in our study to implement counting tasks are feedforward, Elman and ESCN.

2. Experimental Setup

In this study, a Khepera Robot simulator, YAKS (Yet Another Khepera Simulator) is used for accomplishing memorization and counting tasks defined. In YAKS, the sensor values are based on real readings from the KHEPERA sensors (Khepera Referece Manual, 2002), making it easy to transfer the simulated controller to the real robot. The sensors of the two-wheeled Khepera robot used in the simulator are shown in Figure 8.

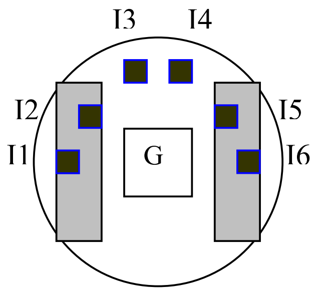

Figure 8. Sensors of the Khepara robot used in this study: **I1, I2....I6**: infrared sensors; **G**: ground (zone) sensor.

The tasks are implemented using zone sensors instead of light sensors to be able to obtain a good subset of simpler yet complex tasks to achieve "counting" step by step.

The genetic algorithms used for the experiments have the following properties:

- a population size of 200 individuals,
- truncation selection: deterministic choice of the best 25 individuals, from a set of 200 offspring and 25 parents,
- binary representation of the genotype,
- no crossover,
- bit-wise mutation,
- and the initial population is generated randomly.

As a first case, a T-maze task was implemented by using a zone sensor (Figure 9). In this case, the T-maze task was completed successfully with evolving a feed-forward neural net. The robot turns either right or left at the junction depending on the presence of a zone.

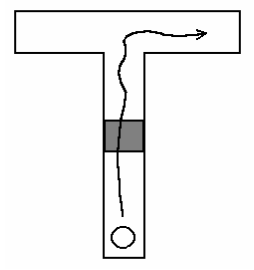

Figure 9. T-maze environment with a zone sensor instead of a light sensor.

Figure 10 shows the maze used for counting to three. The task for the robot is to enter the third corridor. Because the resulting trajectory will be almost the same for every successful run, one might wonder whether this task involves counting or even memory at all. Though, by inspecting the requirements more closely, we will argue that a succesful controller really needs some kind of memory and shows a behaviour which could be described as counting. In what follows, a certain memory content is interpreted as the state of a neural network, i.e. the internal values that determine the mapping from inputs to outputs. Consider one robot at two different stages in the task: one at position B and the other at position C. The correct behaviour in C is to take the first corridor it encounters on the right, which is clearly different from the behaviour in B which should skip the first corridor and take the second one. As the inputs to the robot are the same in both situations (the sensors have a limited range), we have to conclude that the robot's mapping from inputs to outputs in C differs from that in B. The same reasoning applies to a robot in position A. This means that the robot's state or memory content differs in these three cases. Moreover, because of the fact the different state transitions are triggered by the same input patterns, i.e. the alteration between being in a zone and not being in a zone, this behaviour can be considered as counting.

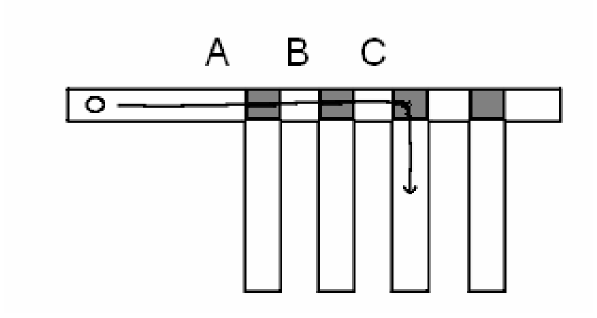

Figure 10. Counting to three task.

2.1 The Elman Network results

The number of hidden nodes used in the Elman network is five. The seven sensors of the simulated robot are mapped on the seven input nodes of the network. The two output nodes are used to set the motor speeds of the robot. The fitness function increases linearly in the horizontal corridor between 0 and 1 from the left to the third corridor and decreases again when going further to the right. Bouncing into walls is discouraged and the fitness is computed incrementaly at each time step. When entering the third corridor a bonus of 10 is given and the fitness keeps increasing when moving further in this corridor. The succes rate of the best individual of generation 400 for 10 evolutionary runs is shown in the following table as percentages.

1	2	3	4	5	6	7	8	9	10
67.0	55.2	20.7	60.5	31.1	43.3	19.1	0.0	20.8	34.3

2.2 The Extended Sequential Cascaded Network

The same experimental setup as before but using an ESCN, produced the following data.:

1	2	3	4	5	6	7	8	9	10
51.5	26.3	0.0	30.7	66.0	20.4	37.6	29.6	51.3	0.0

2.3 Elman network with a different fitness function

In this experiment differs from the one previously described in the following ways. It gives a greater fitness in the lower part of the horizontal corridor. Robots are allowed to slide along walls and the fitness is only evaluated at the end of a run. Figure 11 contains the evolution of the fitness of the best individual of the best run out of 10. The corresponding success rates of these three runs are 90.8, 70.4 and 43.8 % respectively.

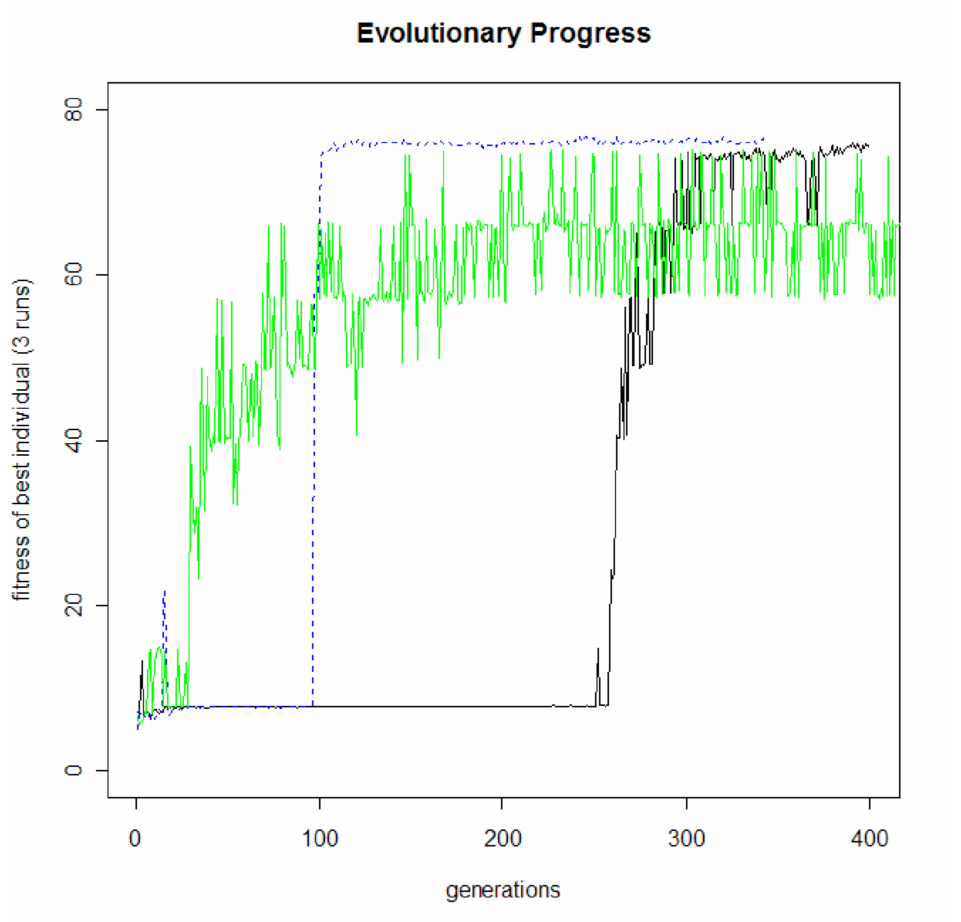

Figure 11. The evolution of the best fitness using an Elman Network.

2.3 A special purpose network

Another test has been conducted which only used the ground sensor as input to 7 hidden, recurrent nodes. The hidden nodes as well as the other input nodes were fully connected to the output nodes. This did not result in a successful, robust behaviour.

Conclusion

The work described in this paper aimed at evolving neural network robot controller for tasks which involve memorization and counting. Several neural networks architectures have been tried to solve the task of always taking the third corridor. For the first fitness function, we evaluated two different NN-controllers with various success in 10 evolutionary runs each. The highest success rate we obtained, was 67.0%. We tested also an alternative fitness function and using the Elman network. In this case we obtained success rates between 43.8% and 90.8%, however we have tested the experimental setup only in three evolutionary runs.
Finally, we cannot conclude whether or not a real "counting to *three*"-task is achieved since we have not analysed the evolved controllers yet.

Acknowledgement
Special tanks to Tom Ziemke for excellent guidance.

References

Ulbricht C. (1996) Handling time-warped sequences with neural networks. In Maes P., et al. (eds.), *From Animals to Animats 4*, Cambridge, MA: MIT Press.
Elman, J. L. (1990) Finding structure in time. *Cognitive Science*, *14*(2), 179-211.
Jakobi, N. (1997a) Evolutionary robotics and the radical envelope of noise hypothesis. *Adaptive Behavior*, *6*.
Jakobi, N. (1997b) Half-baked, ad hoc, and noisy: minimal simulations for evolutionary robotics. In P. Husbands & I. Harvey (eds.), *Fourth European Conference on Artificial Life*, 348-357, MIT Press.
Rylatt, R. M. & Czarnecki, C. A. (2000). Embedding connectionist autonomous agents in time: The 'road sign problem'. *Neural Processing Letters*, *12*, 145-158.
Linåker, F. & Jacobsson, 2001
Thieme, M., Intelligence without hesitation, Master Thesis, Department of Computer Science, University of Skövde, Sweden, 2002.
Bergfeldt, N. and Linåker, F. (2002) Self-organized modulation of a neural robot controller. *International Joint Conference on Neural Networks*.
Z. Reynikova and B. Ryabko, 2000, Using Information Theory Approach to study the communication System and Numerical Competence in Ants.
Khepera Referece Manual, 2002
M. Boden, H. Jacobsson and T. Ziemke, Evolving context-free language predictors, GECCO2000.
Jakobi and Quinn (1998)